dLETTER

The Newsletter for dBase II and III users. Volume 1 Issue 3.

dGenerate dGenerating Your own Reports

By Matt Whelan

The biggest failing dBase has, in the eyes of many users, is its report generator. It has several distinct limitations - you can't do multi-file reports, for example. But the big killer to most people is the restriction to across-the-page report formatting.

It seems any useful database has too much information to fit on one printer line - especially if you have an 80-column printer! Even wide printers set to condensed type can't handle many of the reports we would like to do.

So one of the first questions a dBase user will ask of friends or the dealer is "Can I do multi-line reports? You know, name and address details in one column, so we can put 120 or so characters of information into one 35-character report field?"

The answer they invariably get is no. What a shame...such a wondrous handler of data, which does everything they want, can't show them the information they need in a straightforward manner. And when you think about it, getting the information back out is more important than putting it in - isn't that what data management is all about?

So dBase is only three-quarters of a database package; it's like a car with no wheels. You can rev the engine and change gears, but you sure as hell can't go anywhere...

Ratliff To The Rescue

Fortunately, dBase author Wayne Ratliff made up for any deficiencies in his package by providing a powerful programming language with it. If dBase can't do it for you, you can do it yourself.

What's that? You don't want to write programs? Shame on you! Oh, you don't mind writing programs, but object to having to do so for something as basic to data management as a simple report?

Well, thousands of dBase users have had to write hundreds of thousands of custom report programs to get dBase to do what they wanted. Unfortunately, probably around 90 per cent of those programs were wasted work.

Partially-documented and undocumented features of the report generator make it significantly more powerful than most people ever believed possible. Multi-line reports are a breeze!

Who Has The Answer?

A number of dBase users already know the technique. It has been published in Robert Byers' book "dBase II For Every Business" and I've seen a lift of the essence of Byers' report chapter in an American magazine.

Yet less than one per cent of the people who have attended my dBase programming course have known about it, so I thought it was time we made it more public.

```
STRUCTURE FOR FILE:   B:CLUBS    .DBF
NUMBER OF RECORDS:    00003
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH    DEC
001      CLUB           C     035
002      CONTACT        C     030
003      ADDR1          C     035
004      ADDR2          C     030
005      SUBURB         C     020
006      PCODE          C     004
007      PHONE          C     012
008      MEETINGS       C     090
009      FORMACHIN      C     025
** TOTAL **                 00282
```

Figure 1. Structure of our test file.

Let's look at a typical situation. Our mythical CLUBS database (Figure 1) has a record size of 282 characters - even without the 90-character MEETINGS field it wouldn't fit on most printers.

Our first attempt at a report (Figure 2) lets us get information from only four of the nine fields, and even then we have to make our report field widths narrower than the file fields to get it in.

Fortunately, as we have probably all seen, dBase has the ability to 'wrap' a field, so our address field ends up on two lines (as does one of the contact fields). This is a useful feature, if untidy at times.

Figure 2b is the report .FRM file generated by dBase when we first specified the report. Note that in dBase II this is stored as an ASCII text file and can be edited (see box).

Getting Adventurous

The fact that dBase will 'wrap' fields for us if they are wider than we have allowed for should give us a clue for improved reporting: if, instead of specifying a 35-character string for our first report field (CLUB) we specify a 100-character string (using the '+' sign to 'concatenate' several fields into one), we know it will be broken up roughly into three lines of 35,35 and 30 characters.

If we pick fields of suitable length, we'll get them split at the right place to be on separate lines!

We've used this technique in the report shown as Figure 3. Because the CLUB and ADDR1 fields are 35 characters each, and we've specified a report column width of 35, they and the ADDR2 field get spread neatly over three lines.

Note also the heading specifications in this report form - we've made it left-justified by using the '<' symbol at the start of the heading. We could have right-justified it by preceding it with '>'.

Now We're Warmed Up...

This technique is fine, but it depends on us having fields of similar length. If we wanted to put CLUB+SUBURB+CONTACT in the first column it would probably wrap in the middle of the contact field (the fields are 35, 20 and 30 characters respectively).

Our ability to do multi-line reports would depend heavily on the database structure if this method was our last resort. Also, it would be nice to have a method of putting headings down the side of the page instead of across the top if we are going to have several different fields on separate lines within one column.

Is there something else we can do?

Yes, there is. It all relates to an unusual side-effect of the provision for use of the semi-colon as a line-continuation marker in program files (I assume it's a side-effect; if it's intentional it would have been in the documentation. Wouldn't it?).

If you have a program line that is longer than the screen width, dBase allows you to put in a semi-colon and continue the command on the next line. For example:

```
IF !(yesno)="Y" .AND. .NOT. EOF .AND. ;
ready=0 .AND. &mfield=mname
```

The side-effect is that when you put a semi-colon inside text output, it displays as a line-break! dBase II users who have put semi-colons in their data fields will have seen this - if you haven't, try the following:

```
. store "matt;whelan;was;here" to mname
matt;whelan;was;here
. ? mname
        matt
      whelan
         was
        here
. ? mname+mname
                matt
              whelan
                 was
              herematt
              whelan
                 was
                here
```

PAGE NO. 00001

Simple Report Form In "Normal" Format

Club Name	Contact	Address 1	Suburb
NSW Micro Midgets	V Short	Lower Ground Floor	Anywhere
dBased Users Of Australia	Mrs Dot Prompt	The Highest Level, Suite 1	Buggabill
CompuPro Busblasters Society	The dLetter Crew	Just up the Epsom Road	Waterloo

Figure 2a. A by-the-book report.

```
m=0,l=60,w=78
y
Simple Report Form In "Normal" Format
y
n
35,club
Club Name
15,contact
Contact
15,addr1
Address 1
10,Suburb
Suburb
```

Figure 2b. The 'standard' dBase Report Form file.

Ashton-Tate, We Hate You!

The techniques for doing multi-line reports have remained undocumented by Ashton-Tate for too long! It really annoys me that we have been able to buy the information from Ashton-Tate's book publishing division for something like two years, yet it still isn't in the manuals.

It is ironic (I won't even think about a more cynical description) that a company which was able to produce such bad manuals (dBase II) was at the same time able to publish and sell books as good and as informative as those of Robert Byers.

Why couldn't the people writing the manuals read the company's own books, we wondered?

It only annoyed me a little - until I noticed the credits in the dBase III manual. Robert Byers is its managing editor...aaarrgh!

Management Technology Education Pty Ltd.

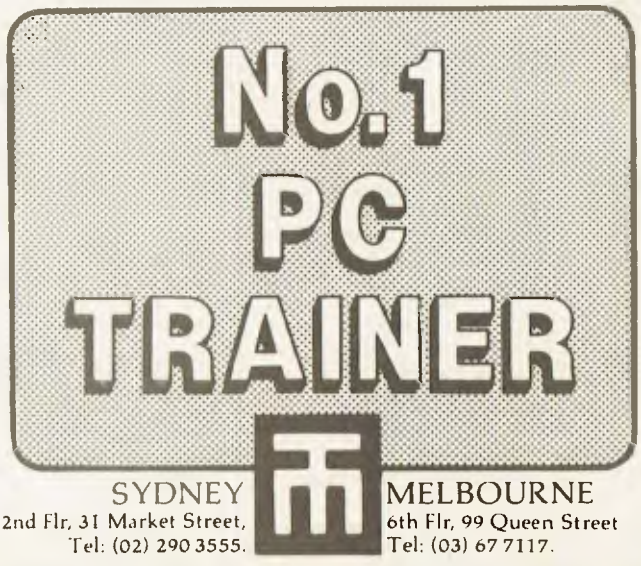
MTE is Australia's leading supplier of quality PC training to business and government.

MTE courses feature top-line instructors, "hands-on" format, professional design, premium quality training facilities, Melbourne and Sydney City Centre locations.

When you have the training need, we have the course.

- PC's as a Business Tool
- Multiplan Introduction & Advanced
- dBASE II & III Intro., Intermediate & Advanced
- WordStar, Introduction & Getting More From WordStar?
- LOTUS 1-2-3 & SYMPHONY

For full details of the above and other MTE PC Courses phone for our **FREE** comprehensive booklet or complete and send the coupon opposite.



SYDNEY
2nd Flr, 31 Market Street,
Tel: (02) 290 3555.

MELBOURNE
6th Flr, 99 Queen Street
Tel: (03) 67 7117.

By the way; if this is happening to you when you don't want it - for example, when an operator has entered semi-colons in addresses or similar - you can cancel this effect with the SET RAW ON command.

If you take advantage of this side effect in report, you can have total control over line-breaks. The BIG thing to remember is that the semi-colon has to be output as text!

That means that within a field specification it has to be enclosed in quotes - if not, dBase will treat it as the line continuation character and expect the field specification to continue on the next line.

And, if you use it within headings (you want multi-line headings, with underlining, don't you?) it doesn't need the quotes - dBase considers the headings to be text by default.

Look at the report in Figure 4. Bliss! We have all our information there, in 78 columns, with individual field headings and all. Neat, tidy, effective.

How do we get the headings down the first column? There's no rule that says a report field specification has to contain a database field; in this case we just put the headings in - all enclosed in quotes to show it is text - separated by semi-colons to force line breaks. Then we put the matching fields

dBase III Reports

The report forms shown here are dBase II .FRM files, and they WON'T work as shown for dBase III. However, that's only because the .FRM file is created, stored and modified differently in dBase III.

The techniques described will work equally well for dBase III if entered into the Field Specification in the full-screen report form editor.

Watch out, though. Helpful dBase will put a default field width in for you, and it will count all characters in the field specification as if they were going to appear on one line. You must remember to over-ride the width dBase fills in for you.

As long as you watch out for this, you can do all the things listed here as well, or better, with dBase III.

```
PAGE NO. 00001

                    Second Report Using Joined Fields

Club Name and Street Address        Contact/Suburb/Postcode

NSW Micro Midgets                   V Short
Lower Ground Floor                  Anywhere            2001
123 Any Street

dBased Users Of Australia           Mrs Dot Prompt
The Highest Level, Suite 1          Buggabilla          3456
123 Mortein Drive

CompuPro Busblasters Society        The dLetter Crew
Just up the Epsom Road              Waterloo            2017
from
```

Figure 3a. Wrapping fields into multi-line format.

```
m=0,l=60,w=78
y
Second Report Using Joined Fields
y
n
35,club+addr1+addr2
<Club Name and Street Address
30,contact+suburb+pcode
<Contact/Suburb/Postcode
```

Figure 3b. The report form - simple, huh?

No. 1 dBASE TRAINER

SYDNEY
2nd Flr, 31 Market Street,
Tel: (02) 290 3555.

MELBOURNE
6th Flr, 99 Queen Street
Tel: (03) 67 7117.

Management Technology Education Pty Ltd.

When you need dBASE training call MTE, Australia's largest supplier of quality PC training.

dBASE courses include:

- dBASE II and dBASE III Introduction
- dBASE II and dBASE III Intermediate
- dBASE Advanced with Les Bell

SEND NOW FOR YOUR **FREE** COMPREHENSIVE COURSE BOOKLET

☐ Please send me FREE the MTE comprehensive PC course booklet.

Name: .....

Position: .....

Company: .....

Address: .....

..... Postcode: ..... Phone: .....

into report column 2, again separated by semi-colons. Presto!

(Note: we specified a field width of 35 characters for the second column, so we don't need a semi-colon after the 35-character database fields; dBase will automatically wrap at that point, and if we had a semi-colon there we would end up with an extra blank line.)

Notice the extra spacing between each record in this report - there's four blank lines between records even though dBase only allows for double spacing. Looking at Figure 4b, the contents of the report form, you can see how that was done (if you hadn't already guessed) using our friendly semi-colon again.

Two-File Reports

Just as any multi-file operation in dBase is limited, two-file reports are possible but in most cases impractical.

However, if you have 'linked' files - normally used only where the information should be put in one file, but there aren't enough fields available - you can report on both at once using the technique shown in Figure 5.

The dBase report generator isn't limited to the current work area's file - it can reference the other work area,

Editing dForms

dBase Report form files are stored on disk as straight ASCII text, so they can be edited using MODIFY COMMAND or your word processor (non-document mode, please).

The only thing you have to watch out for (apart from normal syntax considerations, of course) is maintaining the report form's structure. For example, if you want to take out a field heading, you can't just delete the line - you have to leave a blank line in its place, because dBase expects either a heading or an empty line to indicate no heading.

Similarly you have to be careful if you change the yes/no answers to questions like "Page Heading?", "Are Totals Required?" and so on. For example, if you say yes to totals further answers are required (on subtotals and on each numeric field in the report).

Thus the report form layout will use a different number of lines depending on your answers - get that wrong and dBase will object loudly.

The next problem is the totally unhelpful error messages you get if you make a mistake in a report form. When entering the report interactively, dBase will tell you if a field specification is incorrect - if you edit the report form you get no such help. You can only search carefully for your syntax error, or start again...

```
PAGE NO. 00001

                    Using Semi-colons to Force Line Breaks

CLUB:       NSW Micro Midgets                  MEETINGS: This club holds
CONTACT:    V Short                            meetings every time the
ADDR1:      Lower Ground Floor                 members can get together,
ADDR2:      123 Any Street                     which isn't often...
SUBURB:     Anywhere            2001
PHONE:      (02) 3456789
MACHINE:    CP/M and Tiny BASIC


CLUB:       dBased Users Of Australia          MEETINGS: Buggabilla
CONTACT:    Mrs Dot Prompt                     Psychiatric Hospital and Home
ADDR1:      The Highest Level, Suite 1         for the Aged and Demented.
ADDR2:      123 Mortein Drive                  Every day, from sunup.
SUBURB:     Buggabilla          3456
PHONE:      Disconnected
MACHINE:    You name it, we'll kill..


CLUB:       CompuPro Busblasters Society       MEETINGS: Pre-dawn on Sundays
CONTACT:    The dLetter Crew                   public holidays and any other
ADDR1:      Just up the Epsom Road             inconvenient time.
ADDR2:      from
SUBURB:     Waterloo            2017
PHONE:      987654321
MACHINE:    Viasyn (not the medicine)
```

Figure 4a. Taking full advantage of Report's capabilities.

```
m=0,l=60,w=78
y
Using Semi-colons to Force Line Breaks;=============================
y
n
10,"CLUB:;CONTACT:;ADDR1:;ADDR2:;SUBURB:;PHONE:;MACHINE:"

35,club+contact+';'+addr1+addr2+';'+suburb+pcode+';'+phone+';'+formachin+';;'

30,"MEETINGS: "+meetings
```

Figure 4b. The semi-colon in action.

```
. select secondary
. use file2
. sele prim
. set linkage on
. report form test4


PAGE NO. 00001

                                Testing Two-file Report

Club Details                            Contact plus file2 info

NSW Micro Midgets                       V Short
Lower Ground Floor                      Hello there from    file2
123 Any Street
Anywhere            2001

dBased Users Of Australia               Mrs Dot Prompt
The Highest Level, Suite 1              here's record2 in   file2
123 Mortein Drive
Buggabilla          3456

CompuPro Busblasters Society            The dLetter Crew
Just up the Epsom Road                  here's record3 in file2
from
Waterloo            2017
```

Figure 5a. Limited multi-file reporting is available.

```
m=0,l=60,w=78
y
testing two-file report
y
n
35,club+addr1+addr2+';'+suburb+pcode
<Club Details
30,contact+something+else
<Contact plus file2 info
```

Figure 5b. The fields SOMETHING and ELSE are in the second file.

memory variables, and so on. And field specifications can just as easily be numeric or string expressions. This opens up a lot more useful possibilities.

How many people have a database containing fields for purchase price, sales tax, margin, and retail price? Or any similar collection where one or more fields (here retail price and sales tax) is a product of the others. They usually update the files regularly with commands like:

```
REPLACE ALL stax WITH cost*0.20
REPLACE ALL retail WITH cost*margin+stax
```

Who needs sales tax and retail price fields in the database anyway it's just a waste of space. You can calculate the prices afresh every time you do the report by putting field specifications like this in your report form:

```
5,cost*0.20
<TAX
7,cost*margin+cost*0.20
<RETAIL
```

Do your prices vary with currency fluctuations? Your report form could reference a memory variable containing today's exchange rate.

What about database enquiries? Perhaps you work from the screen as often as from a printed price list? Well, a screen format file can contain the same calculations as the report form...

dBriefing

Feedback and Food for Thought

Dear dLetter,

Your comment on dBase books omits three titles I found far superior to Adam Green's 'dBase II User's Guide' which leaves too many questions unanswered. Green's book is, in fact, so simplistic that it just gets people into trouble without any inkling as to why, or how to get out of it, for example, how to delete a file.

The titles I would recommend (having taught myself without any prior computer experience) are:

* **Rob Krumm: 'Understanding & Using dBase II'**, Bradley Communications, Prentice Hall.

At $31.50 this book is not the most expensive. It is well indexed, very comprehensive and covers some of the advanced areas which the beginner at least needs to refer to. One of its best features is the way in which new developments in a topic are underlined in the relevant accompanying programs. The text is fully detailed without the annoying anecdotal verbiage of people like Green and Byers. Creating pages, page headings, page numbers, totals, control breaks, subtotals and averages are explained very clearly, together with reasonable references to functions, printers and the transfer of information between dBase II and other packages.

* **M. dePace: 'Working with dBase II'**, Grenada.

At only $14.95 (and being readily available from Tandy) this book is a must for any dBase user. (Tandy got 20 copies to me in Burnie, Tasmania, in a matter of two days.)

While the section on programming is not quite as clear as Krumm, the chapters on macro substitution, memory variables, data mainpulation, functions, set commands and special techniques are second to none. The index is average but is supplemented by three excellent appendices including a summary of dBase commands - listed in a way they can be found quickly - unlike the manual.

Both these books develop a business system which covers most principles and which can be adapted to other applications.

* **McMahon, Hoove & Popp: 'Report Writing in dBase II'**, Spectrum, $25.25.

This is a good book for those who feel apprehensive about programming. It brings home how completely inadequate the dBase manual is in its coverage of the dBase report generator, while Byers' & Green's are equally culpable - particularly when you look at their cost.

One risk in using 'Report Writing in dBase II' is that committed anti-programmers soon find themselves seduced into writing programs. The book is a very useful transition as well as being absolutely essential for anyone who wants to make full and enlightened use of the report generator. The only annoying aspect of this book is a modicum of anecdotal garbage about running a restaurant and the confusing chapter headings which follow this theme throughout.

As a postscript, Spectrum Books also publishes '101 Questions about dBase II' by Ing and Fletcher. This is an odd hotchpotch of padding, trivia and vital things which nobody else bothers to tell you. At $31.50 it's well worth borrowing from a friend or getting the local library to buy and just noting the things which are a complete revelation to the distraught fossicker.

Ted Bolton
Burnie, Tasmania.

Thanks for the comments. Everyone has different selection criteria when looking for useful books - some of the features you don't like are the things which attract me to particular books.

My all-time favourite remains the 'Advanced Programmer's Guide' by Castro et al (mentioned in the first issue of dLetter). If I could only have one dBase book, that'd have to be it.

Dear dLetter,

I have been using dBase II for over a year now and am delighted with it. I am very pleased to see dLetter and look forward to further issues.

Basically, I am a hardware hacker who has a few practical uses for dBase. My hobbies are veteran and vintage T-model Fords plus personal computing. I got the job of keeping our club records, membership and car lists plus printing the labels for the monthly magazine. I have also put my garage 'online' with a stock control system in dBase, which works well.

My problem is when I come to print out a register of members plus lists of cars. The database is in two parts - member info and car info.

When I attempt to print out the combined list I cannot - other than by manual means - make a neat listing. I cannot skip to a new page after a set number of lines, as the number of lines per member varies with the number of car entries. I want to ensure that one member's details and list of cars is never broken over two pages. I suppose this should use line counting somehow, but I'm damned if I can figure it out.

I take exception to Les Bell's statements about SORT as I got dSort in my package of 2.41 and it is truly magic

· really fast. However, I do support him if the original SORT from dBase is used. Keep up the good work.

G. Hawthorn
Glen Waverley, Vic.

A line-counting routine will do the trick for you. The accompanying listing shows such a line counting routine, using your example as its basis.

With two databases set up as you suggest, the first database has a one-to-many relationship to the second; that is, one owner may have many cars, so a record in the MEMBERS database may have multiple related records in the CARS database.

This routine can be adapted for any situation where you're unsure of how many records in a second database may relate to a record in the first. The routine is commented, but a few extra words might help.

My sample files are very simple structurally. MEMBERS.DBF basically consists of:

FIRSTNAME	C	015
LASTNAME	C	015
MEMNO	N	003

CARS.DBF is:

CAR	C	030
DETAILS	C	100
MEMNO	N	003

Say Jane Bloggs has a membership number of 47. She owns three cars which are each listed on a separate record in CARS.DBF, with the matching value of 47 in the MEMNO field of each record.

CARS.CMD does most of the work - opening files, initialising variables, processing the MEMBERS database and finding the matching CARS records.

LCOUNT.CMD takes care of all the page break activity. The line:

"COUNT TO numlines FOR memno=mno"

lets us 'look ahead' to see how many lines we'll have to print for the current member. If this number will take us past the end of the page, we force a page break, print a heading and reset our line counter. I've set a page length of 55. Experiment to find what you need.

This is not necessarily the best solution to the problem (it is, in fact, the first thing that came into my head). For a start, it's slow, with a noticeable pause between printing each member's records. Also, I haven't checked for any exceptions, such as a member having no cars (they'll be placed on the printout anyway using this program). But it should give you the general approach.

The most important part is the technique of establishing a line counter, counting ahead the number of lines needed for each member and checking this against your established page length. This technique should be adaptable to a variety of situations.

```
*CARS.CMD
*Lists selected membership details from primary database and
*details of each member's cars from the secondary database.
*MEMBERS.DBF and CARS.DBF have MEMNO as the matching field.

*Initialise variables and environment
SET TALK OFF
STORE "THIS IS THE HEADING" to heading
STORE "Name                          Cars" TO subhead
STORE "                              " TO spaces

*Storing 60 to the line counter will force a heading at beginning
*of first page.
STORE 60 TO lcount
STORE 0 TO numlines

*Open databases
SELE PRIM
USE members INDEX memno
SELE SECO
USE cars INDEX cmemno

*Select primary database, position pointer at beginning of file
SELE PRIM
GO TOP
SET PRINT ON

*Main loop to process MEMBERS file
DO WHILE .NOT. EOF

   *Store MEMNO to variables so matching membership numbers can be
   *found in the CARS database.
   *Use CARS database
   STORE memno TO mno
   STORE STR(memno,3) TO mnos
   SELE SECO

   *Call page break routine
   DO lcount

   *Print details from MEMBERS database
   SELE PRIM
   ? TRIM(firstname)+" "+lastname

   *Find first matching record in CARS database
   *and then print all the matching records.
   SELE SECO
   FIND &MNOS
   DO WHILE memno = mno .AND. .NOT. EOF
      ? spaces+car+details
      SKIP
   ENDDO
   ?

   *Increment line counter by number of records printed (numlines
   *for CARS records + 1 for the MEMBER record) plus 1 line spacing
   *Select MEMBERS database and proceed to next record
   STORE lcount+numlines+2 TO lcount
   SELE PRIM
   SKIP
ENDDO

*Tidy up
EJECT
SET PRINT OFF
SET TALK ON
RETURN
```

Example program to print a listing from two files, allowing page breaks depending on the number of matching records to be printed from the second file.

```
*LCOUNT.CMD
*Line counting routine for many-to-one record printing from two databases.

*Count number of cars belonging to owner x, and store this in variable
*'numlines'.
COUNT TO numlines FOR memno=mno

*If current line count + number of matching records to be printed will
*exceed the page line limit
*   eject
*   print headings
*   reset line counter to initial setting
IF (lcount+numlines) > 55
   EJECT
   ? Heading
   ?
   ?
   ? Subhead
   ?
   STORE 5 TO lcount
ENDIF
RETURN
```

Line counting subroutine, called from CARS.CMD

SAVE $38 ★ SAVE $38 ★ SAVE $38 ★ SAVE $38 ★ SAVE $38 ★ SAVE $38

A Special Offer From *Your Computer*

SUBSCRIBE TO dLETTER NOW
AND GET A YEAR'S SUBSCRIPTION TO *YOUR COMPUTER* FREE!
ALL FOR THE SPECIAL PRICE OF $25

- An exciting new source of information, hints, tips and techniques for dBase users.
- The only independent national newsletter for dBase II and dBase III users.
- Written BY dBase users FOR dBase users. The three editors – Les Bell, Matt Whelan and Rose dVines – have over 10 years of dBase experience tucked beneath their collective belt.
- Full of practical information about dBase and dBase-related products.
- Supported by the resources of *Your Computer* – Australia's leading microcomputer magazine.

The normal subscription price for dLetter is $28; 12 issues of Your Computer would cost you $35.40. You can get both, mailed to your home or office, for the special introductory price of $25!
Just fill in the coupon below to secure your copy, and let us dLiver dLetter to you.

✂ ---

dLETTER SUBSCRIPTION FORM
SPECIAL COMBINED SUBSCRIPTION OFFER:
12 issues dLETTER PLUS 12 issues YOUR COMPUTER for $25

Name: .....

Address: .....

..... Postcode: .....

Phone ( ) .....

Tick box to indicate method of payment:

☐ Bankcard ☐ American Express ☐ Mastercard

☐ Cheque/Money order (payable to Federal Publishing Company)

Credit card number: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiry date: ...../.....

Signature: .....
(unsigned credit card orders cannot be accepted)

Send to: Federal Publishing Company, PO Box 227, Waterloo, NSW 2017.

SAVE $38 ★ SAVE $38 ★ SAVE $38 ★ SAVE $38 ★

dFects

This month's dFect was sent to us by Linda Flanagan, who uses dBase 2.41 under CP/M-86 in her work at Colonial Mutual:

I have found that when I want to evaluate a substring inside an @ x,y SAY, and the substring length is an expression or variable, I can't. dBase makes the length 1 no matter what the expression or variable is.

I have found the only way to get around this is to evaluate the substring first, storing it in yet another variable, and then 'SAYing' the variable.

For example, I had a 50-character address string to work with which contained the FULL address in the format:

street, suburb, state, postcode

This then needed to be split into three address lines for printing on letters.

Suppose we have a field rep:addrss containing the following data:

2 THE AVENUE,NUNAWADING,VICTORIA 3131

and the variable 'lena' is equal to the length of rep:addrss; that is, lena = LEN(rep:addrss). Program 1 will produce:

```
2
N
V
```

while Program 2 will produce the desired result:

```
2 THE AVENUE,
NUNAWADING,
VICTORIA 3131
```

PROGRAM 1

```
*Locate the comma indicating the end of the street portion of the
*address, and store the comma's position in a variable
STORE @(",",rep:addrss) TO pos1

*Separate the street from the remainder of the address, print it,
*and then separate the suburb from the state and postcode
IF pos1 > 0
   @ 14,15 SAY $(rep:addrss,1,pos1)
   STORE $(rep:addrss,pos1+1,lena-pos1-1) TO rest
   STORE @(",",rest) TO pos2

*Print the suburb and then the state and postcode
   IF pos2 > 0
      @ 15,15 SAY $(rep:addrss,pos1+1,pos2)
      @ 16,15 SAY $(rep:addrss,pos1+pos2+1,lena-pos1-pos2-1)
   ELSE

*If there are only two parts to the address (ie. no state and
*postcode, print the second part
      @ 15,15 SAY rest
   ENDIF
ELSE

*If the address is a single line (no commas) print it all on one
*line
   @ 14,15 SAY rep:addrss
ENDIF
```

PROGRAM 2

```
STORE @(",",rep:addrss) TO pos1
IF pos1 > 0

*Store first line to separate variable and print that variable
   STORE $(rep:addrss,1,pos1) TO addr:line1
   @ 14,15 SAY addr:line1
   STORE $(rep:addrss,pos1+1,lena-pos1-1) TO rest
   STORE @(",",rest) TO pos2
   IF pos2 > 0
      STORE $(rep:addrss,pos1+1,pos2) TO addr:line2
      @ 15,15 SAY addr:line2
      STORE $(rep:addrss,pos1+pos2+1,lena-pos1-pos2-1) TO addr:line3
      @ 16,15 SAY addr:line3
   ELSE
      @ 15,15 SAY rest
   ENDIF
ELSE
   @ 14,15 SAY rep:addrss
ENDIF
```

dFunct

The dBase Function of the Month

By Rose dVines

The lower-to-upper-case function (!() In dBase II and UPPER() in dBase III) is one of the simplest functions to use. It is invaluable in testing the validity of string input and allowing for a bit of leeway in the operator's input.

One use of the upper-case function is to test the operator's responses to a screen enquiry. Suppose we want the operator to reply to a prompt such as:

Do you want to add this record (Y/N)?

The following piece of code:

```
STORE " " TO reply
@ 1,0, SAY "Do you want to delete this ;
record (Y/N)?" GET reply
STORE !(reply) TO reply
* reply=UPPER(reply) in dBase III
READ NOUPDATE
```

will result in the appropriate action being taken, no matter whether the response is "Y", "y", "N" or "n".

Another use of the upper-case function is in searching files for specific records. Say a database contains a field "lastname". Data entered in the field is likely to come in a variety of combinations of upper and lower case when typed in by the operators; ie.

SMITH
Smith and
smith

If you INDEX on !(lastname) / UPPER(lastname), the operator need only search for 'SMITH' to ensure finding any of the above alternatives. If you keep this as a rule when INDEXing, you'll be sure of more success in data retrieval.

A similar approach can be used with the LOCATE command:

LOCATE FOR !(lastname)='SMITH' (II)

LOCATE FOR UPPER(lastname)='SMITH' (III)

One case where there is an alternative to using UPPER() in dBase III is with the SORT command. Using the option 'SORT /C' will ignore the case of strings while SORTing.

dBase III also has the reverse function - LOWER(). It does what you'd expect. It can be used in the same way as UPPER() - although for consistent data retrieval it's probably best to stick to one function or the other. It can also be used in conjunction with UPPER() to ensure surnames (and suchlike) are printed with the first letter capitalised and the rest in lower case.

dLetter is published monthly by the Federal Publishing Company Pty Ltd. Printed by ESN - The Litho Centre, Waterloo 2017. All correspondence should be addressed to: Rose dVines, Federal Publishing Company, PO Box 21, Waterloo 2017; phone (02) 663 9999.